o escopo do trabalho é reduzido. O cronograma de entrega nunca é estendido. No entanto, isso pode causar problemas, pois significa que os planos do cliente podem ser afetados. Reduzir o escopo pode criar mais trabalho para os clientes se eles tiverem que usar um sistema incompleto ou modificar a sua maneira de trabalhar entre um lançamento do sistema e outro.

Uma grande dificuldade no planejamento ágil é que ele se baseia no envolvimento e na disponibilidade do cliente. Esse envolvimento pode ser difícil de organizar, já que os representantes do cliente às vezes têm de priorizar outro trabalho e não estão disponíveis para o jogo do planejamento. Além disso, alguns clientes podem estar mais familiarizados com os planos de projeto tradicionais e podem achar difícil se envolver em um processo de planejamento ágil.

O planejamento ágil funciona bem com times de desenvolvimento pequenos e estáveis, que podem se reunir e discutir as histórias a serem implementadas. No entanto, nas situações em que os times são grandes e/ou estão geograficamente distribuídos, ou quando os membros dos times mudam frequentemente, é praticamente impossível que alguém se envolva no planejamento colaborativo que é essencial para o gerenciamento de projetos ágeis. Consequentemente, os projetos grandes normalmente são planejados usando abordagens tradicionais para o gerenciamento de projetos.

23.5 TÉCNICAS DE ESTIMATIVA

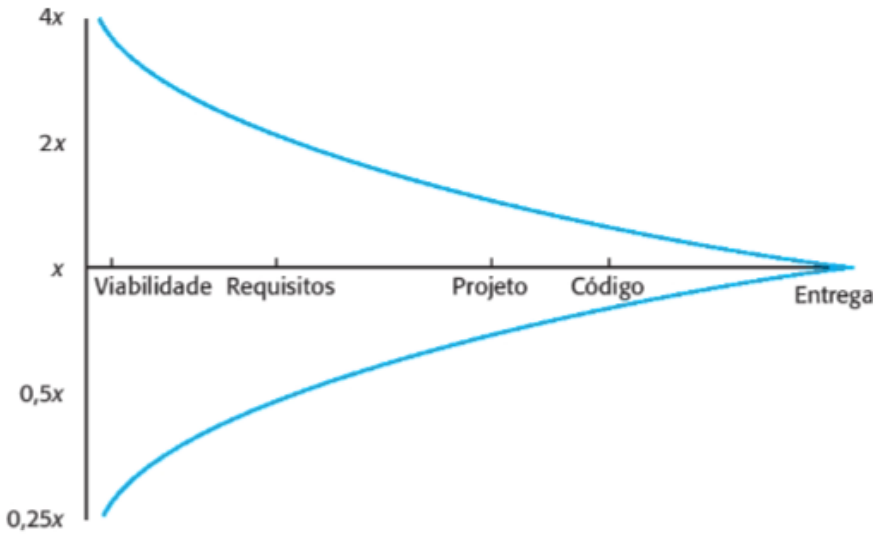
É difícil estimar cronogramas de projeto. É preciso fazer estimativas iniciais com base em uma definição incompleta dos requisitos de usuário. O software pode ter de rodar em plataformas desconhecidas ou usar novas tecnologias de desenvolvimento. As pessoas envolvidas no projeto e suas habilidades provavelmente não serão conhecidas. Existem tantas incertezas que é impossível estimar com precisão os custos de desenvolvimento do sistema durante os estágios iniciais de um projeto. Contudo, as organizações precisam fazer estimativas de esforço e custo de software. Há dois tipos de técnicas que podem ser utilizadas para fazer estimativas:

1. *Técnicas baseadas na experiência*. A estimativa dos futuros requisitos de esforço baseia-se na experiência do gerente em projetos anteriores e na área da aplicação. Essencialmente, o gerente faz um julgamento fundamentado de quais deverão ser os requisitos de esforço.
2. *Modelagem algorítmica do custo*. Uma abordagem usando fórmulas é usada para calcular o esforço do projeto, com base nas estimativas dos atributos do produto, como tamanho, características de processo e experiência do pessoal envolvido.

Em ambos os casos, é preciso usar o bom senso para estimar diretamente o esforço ou as características do projeto e do produto. Na fase inicial de um projeto, essas estimativas têm uma larga margem de erro. Com base nos dados coletados em muitos projetos, Boehm *et al.* (1995) descobriram que as estimativas iniciais variam significativamente. Se a estimativa inicial do esforço necessário for x meses de esforço, eles constataram que o intervalo pode ir de $0{,}25x$ a $4x$ do esforço real, em comparação à medida de quando o sistema foi entregue. Durante o planejamento

do desenvolvimento, as estimativas vão ficando cada vez mais precisas enquanto o projeto avança (Figura 23.9).

FIGURA 23.9 Incerteza da estimativa.



As técnicas baseadas na experiência contam com a experiência do gerente em projetos anteriores e com o esforço real despendido nesses projetos em atividades relacionadas ao desenvolvimento de software. Tipicamente, são identificados os entregáveis a serem produzidos em um projeto e os diferentes componentes de software que devem ser desenvolvidos. Tudo é documentado em uma planilha, estimado individualmente e o esforço total necessário é calculado. Normalmente, é benéfico ter um grupo de pessoas envolvidas na estimativa de esforço e perguntar para cada uma delas a explicação da sua estimativa. Muitas vezes, isso revela fatores que outros não consideraram, e se itera para uma estimativa acordada pelo grupo.

A dificuldade com as técnicas baseadas na experiência é que o projeto de um novo software pode não ter muito em comum com os projetos anteriores. O desenvolvimento de software muda com muita rapidez, e um projeto frequentemente usará técnicas novas para o time, como *web services*, configuração de sistemas de aplicação ou HTML5. Quem não tiver trabalhado com essas técnicas, a experiência prévia pode não ajudar na estimativa do esforço necessário, tornando mais difícil produzir estimativas exatas de custos e de cronograma.

É impossível dizer se as abordagens baseadas na experiência ou em algoritmo são as mais precisas. As estimativas de projeto costumam ser autorrealizáveis. A estimativa é utilizada para definir o orçamento do projeto, e o produto é ajustado para que o orçamento seja cumprido. Um projeto que esteja dentro do orçamento pode ter alcançado isso em detrimento das características implementadas no software desenvolvido.

Para fazer uma comparação da precisão dessas técnicas, uma série de experimentos controlados seria necessária, nos quais várias técnicas seriam utilizadas independentemente para estimar o esforço e os custos do projeto. Nenhuma mudança no projeto seria permitida, e o esforço final poderia ser comparado. O gerente de projeto não conheceria as estimativas de esforço, então não seria introduzido nenhum viés. No entanto, esse cenário é completamente impossível nos projetos reais, de modo que nunca haverá uma comparação objetiva dessas abordagens.

23.5.1 Modelagem algorítmica de custo

A modelagem algorítmica de custo usa uma fórmula matemática para prever os custos do projeto, com base nas estimativas do tamanho do projeto, do tipo de software que está sendo desenvolvido e de outros fatores do time, do processo e do produto. Os modelos algorítmicos de custo são desenvolvidos por meio da análise dos custos e dos atributos de projetos concluídos e, depois disso, pela busca da fórmula mais ajustada para os reais custos incorridos.

Os modelos algorítmicos de custo são utilizados principalmente para fazer estimativas dos custos de desenvolvimento de software. No entanto, Boehm *et al.* (2000) discutem uma série de outros usos para esses modelos, como a preparação das estimativas para os investidores nas empresas de software, estratégias alternativas para ajudar a avaliar os riscos de fundamentar as decisões sobre reúso, redesenvolvimento ou terceirização.

A maioria dos modelos algorítmicos para estimar o esforço em um projeto de software baseia-se em uma fórmula simples, apresentada a seguir:

$$\text{Esforço} = A \times \text{Tamanho}^B \times M$$

A: um fator constante, que depende das práticas organizacionais locais e do tipo de software que está sendo desenvolvido.

Tamanho: uma avaliação do tamanho do código de software ou uma estimativa da funcionalidade expressada em pontos de função ou pontos de aplicação.

B: representa a complexidade do software e normalmente fica entre 1 e 1,5.

M: é um fator que leva em conta os atributos do processo, do produto e do desenvolvimento, como os requisitos de dependabilidade do software e a experiência do time de desenvolvimento. Esses atributos podem aumentar ou diminuir a dificuldade global de desenvolver o sistema.

O número de linhas de código-fonte (SLOC) no sistema entregue é a métrica de tamanho fundamental utilizada em muitos modelos algorítmicos do custo. Para estimar o número de linhas de código em um sistema, é possível usar uma combinação de abordagens:

1. Comparar o sistema a ser desenvolvido com sistemas parecidos e usar seu tamanho de código como base para a estimativa.
2. Estimar o número de pontos de função ou de aplicação no sistema (ver a próxima seção) e, por meio de fórmulas, converter esses pontos em linhas de código na linguagem de programação utilizada.
3. Classificar os componentes do sistema usando o julgamento de seus tamanhos relativos e considerar um componente de referência conhecido para traduzir essa classificação em tamanho de código.

A maioria dos modelos algorítmicos de estimativa tem um componente exponencial (**B** na equação anterior) que aumenta com o tamanho e a complexidade do sistema. Isso reflete o fato de que os custos normalmente não aumentam de modo linear com o tamanho do projeto. À medida que o tamanho e a complexidade do software aumentam, incorrem custos extras em virtude da sobrecarga de comunicação dos times maiores, do gerenciamento de configuração mais complexo, da

integração mais difícil do sistema etc. Quanto mais complexo o sistema, mais esses fatores afetam o custo.

A ideia de usar uma abordagem científica e objetiva para a estimativa do custo é atraente, mas todos os modelos de custo algorítmicos sofrem dois problemas fundamentais:

1. É praticamente impossível estimar o **Tamanho** com precisão em um estágio inicial de um projeto, quando só se tem a especificação. As estimativas de pontos de função e pontos de aplicação (ver adiante) são mais fáceis de produzir do que as estimativas de tamanho do código, mas também costumam ser imprecisas.
2. As estimativas da complexidade e dos fatores do processo que contribuem para **B** e **M** são subjetivas. As estimativas variam de uma pessoa para outra, dependendo da sua formação e experiência no tipo de sistema que está sendo desenvolvido.

A estimativa precisa do tamanho do código é difícil em um estágio inicial de um projeto porque o tamanho do programa final depende das decisões de projeto, que podem não ter sido tomadas quando a estimativa for necessária. Por exemplo, uma aplicação que exija gerenciamento de dados de alto desempenho pode implementar o seu próprio sistema de gerenciamento de dados ou usar um sistema de banco de dados comerciais. Na estimativa inicial do custo, é improvável saber se existe um sistema de banco de dados comercial que tenha desempenho suficiente para satisfazer os requisitos de desempenho. Portanto, não dá para saber quanto código de gerenciamento de dados será incluído no sistema.

A linguagem de programação usada para o desenvolvimento do sistema também afeta o número de linhas de código a serem desenvolvidas. Uma linguagem como Java poderia significar que mais linhas de código são necessárias do que se fosse utilizada a linguagem C (por exemplo). No entanto, esse código extra permite mais checagem em tempo de compilação, então os custos de validação tendem a ser menores. Não está claro como isso deve ser levado em conta no processo de estimativa. O reúso de código também faz diferença, e alguns modelos estimam explicitamente o número de linhas de código reusadas. Entretanto, se sistemas de aplicação ou serviços externos forem reusados, é muito difícil calcular o número de linhas de código-fonte que eles substituem.

Produtividade de software

A produtividade de software é uma estimativa da quantidade média de trabalho de desenvolvimento que os engenheiros de software realizam em uma semana ou em um mês. Portanto, ela é expressa como linhas de código/mês, pontos de função/mês e assim por diante.

No entanto, embora a produtividade possa ser medida facilmente quando há um resultado tangível (por exemplo, um administrador processa *N* despesas de viagens/dia), a produtividade de software é mais difícil de definir. Pessoas diferentes podem implementar a mesma funcionalidade de maneiras diferentes, usando variadas quantidades de linhas de código. A qualidade do código também é importante, mas, até certo ponto, é subjetiva. Portanto, não se consegue comparar a produtividade de cada engenheiro. Só faz sentido usar medidas de produtividade com grupos grandes.

Os modelos algorítmicos de custo são uma maneira sistemática de estimar o esforço necessário para desenvolver um sistema. No entanto, esses modelos são complexos e difíceis de usar. Existem muitos atributos e um escopo considerável para incertezas na estimativa de seus valores. Essa complexidade significa que a aplicação prática da modelagem algorítmica de custo tem se limitado a um número relativamente pequeno de empresas grandes, trabalhando principalmente na engenharia de sistemas de defesa e aeroespaciais.

Outra barreira que desestimula o uso dos modelos algorítmicos é a necessidade de calibração. Os usuários do modelo devem calibrar seu modelo e os valores dos atributos usando seus próprios dados históricos do projeto, pois eles refletem a prática e a experiência local. No entanto, muito poucas organizações coletaram dados suficientes dos projetos passados em uma forma que permita a calibração do modelo. O uso prático dos modelos algorítmicos, portanto, tem de começar com os valores publicados dos parâmetros do modelo. É praticamente impossível um modelador saber o nível de proximidade com que esses dados se relacionam com sua organização.

Ao utilizar um modelo algorítmico de estimativa dos custos, é preciso desenvolver uma série de estimativas (no pior cenário, no cenário esperado e no melhor cenário) em vez de uma estimativa única, além de aplicar a fórmula de estimativa dos custos a todas elas. As estimativas têm mais propensão a serem precisas quando se compreende o tipo de software que está sendo desenvolvido e se calibra o modelo de estimativa de custos usando dados locais ou quando a linguagem de programação e o hardware são predefinidos.

23.6 MODELAGEM DE CUSTOS COCOMO

A mais conhecida técnica e ferramenta de modelagem algorítmica do custo é o modelo COCOMO II. Esse modelo empírico foi derivado pela coleta de dados de muitos projetos de software de diferentes tamanhos. Esses dados foram analisados para descobrir as fórmulas que se encaixavam melhor nas observações. Essas fórmulas vinculavam o tamanho do sistema e fatores do produto, do projeto e do time ao esforço para desenvolver o sistema. O COCOMO II é um modelo disponibilizado gratuitamente, apoiado por ferramentas de código aberto.

O COCOMO II foi desenvolvido com base nos primeiros modelos de estimativa de custos COCOMO (modelagem construtiva de custo, do inglês *Constructive Cost Modeling*), que se baseavam em grande parte no desenvolvimento de código original (BOEHM, 1981; BOEHM; ROYCE, 1989). O modelo COCOMO II leva em conta as modernas abordagens para o desenvolvimento de software, como o desenvolvimento rápido usando linguagens dinâmicas, o desenvolvimento com reúso e a programação de banco de dados. O COCOMO II incorpora vários submodelos baseados nessas técnicas, que produzem estimativas cada vez mais detalhadas.